Title	G関数を用いた数学公式データベースの実装について
Author(s)	森永, 昌義; 甲斐, 博; 野田, 松太郎
Citation	数理解析研究所講究録 (2004), 1395: 205-211
Issue Date	2004-10
URL	http://hdl.handle.net/2433/25950
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録 1395 巻 2004 年 205-211

G関数を用いた数学公式データベースの実装について

森永 昌義
MASAYOSHI MORINAGA
愛媛大学大学院理工学研究科
GRADUATE SCHOOL OF SCIENCE & ENGINEERING, EHIME UNIVERSITY[*]

甲斐 博、野田 松太郎
HIROSHI KAI,MATU-TAROW NODA
愛媛大学工学部
DEPERTMENT OF COMPUTER SCIENCE, EHIME UNIVERSITY[†]

1 はじめに

現在の数式処理の計算は代数的アルゴリズムによる解法を中心としている。しかし、高度な関数の定積分や不定積分のように、代数的アルゴリズムでは容易に計算の行えない数式が数多く存在する。これらのいくらかは、数値計算によっても正しい解を得られない場合もある。より広範な問題に対して数式処理を活用するためには、数学公式データベースを作成し、パターンマッチングを行うことによる方法に頼らざるを得ない。しかし、現在までの数式処理に関する研究では、必ずしもデータベースとの結合は十分ではなく、たとえ結合されていたとしても特定の数式処理システムを強化するという目的で作成されていたもののみであった。そこで、本研究では、数式処理システムの固有の表現や命令形態に依存することなく、数学公式データベースにより記号積分を行えるようなシステムの作成を行うことを目的とする。

本研究では、汎用的なデータベースを作成するため、数式表現には数学的な情報を表現・通信するための標準である OpenMath[1, 2, 3] を用いた。OpenMath による数式表現は XML 形式で表されており、XML の名前が、記号、演算子、関数名の数学オブジェクトである OpenMath オブジェクトに対応する。このような数学オブジェクトを木構造を用いて表現することによって、結果として公式の検索を容易にすることができる。また、記号積分を実行するために、超幾何関数 [6] を一般化した Meijer の G 関数 [4, 5, 7] を実装し、公式の自動生成・検索を行う。G 関数固有な演算によって、G 関数相互間の変換が可能である。この技術を用い、任意の数式処理システムで計算することにより、G 関数を変形し、新しい数学公式を導出し、数学公式データベースに納入することができる。この操作によって大量の数学公式を生成することが可能になる。さらに、新たなインデックス法の提案によりデータベース検索の効率化を実現した。 なお、2·3 章に述べる Meijer の G 関数やデータベース設計に関しては、すでに [10] において述べているが、以後の応用との関連のためにあえて概略を示した。

*morinaga@hpc.cs.ehime-u.ac.jp
†kai@cs.ehime-u.ac.jp,noda@cs.ehime-u.ac.jp

2 Meijer の G 関数

2.1 定義

Meijer の G 関数 [4, 5, 7] は、超幾何関数 $F(\vec{a};\vec{b};z)$[6] を一般化したもので次式によって定義される。

$$G_{pq}^{mn}\left(z\left|\begin{matrix}a_1,\ldots,a_n,a_{n+1},\ldots,a_p\\ b_1,\ldots,b_m,b_{m+1},\ldots,b_q\end{matrix}\right.\right)$$
$$= \frac{1}{2\pi i}\oint_L \frac{\prod_{j=1}^{m}\Gamma(b_j+s)\prod_{j=1}^{n}\Gamma(1-a_j-s)}{\prod_{j=n+1}^{p}\Gamma(a_j+s)\prod_{j=m+1}^{q}\Gamma(1-b_j-s)}z^{-s}ds$$

ここで、$\vec{a}=(a_1,\ldots,a_n),\vec{b}=(a_{n+1},\ldots,a_p),\vec{c}=(b_1,\ldots,b_m),\vec{d}=(b_{m+1},\ldots,b_q)$ とする。
積分路 L は $L_{\gamma+i\infty}$,L_{∞},$L_{-\infty}$ の 3 つのいずれかになる。m,n,p,q は $\vec{a},\vec{b},\vec{c},\vec{d}$ の要素数を示している。

2.2 公式の生成

Meijer の G 関数は次のような関係式によって新たに公式が生成できる。

Shift Operator

$D=\frac{\partial}{\partial z}$ とし、

$$\begin{aligned}A_i &= D+(-a_i+1)\\ B_i &= -D+(b_i-1)\\ C_i &= -D+c_i\\ D_i &= D-d_i\end{aligned}$$

とすると、

$$\begin{aligned}A_iG(\vec{a};\vec{b};\vec{c};\vec{d};z) &= G(\vec{a}-\vec{e_i};\vec{b};\vec{c};\vec{d};z)\\ B_iG(\vec{a};\vec{b};\vec{c};\vec{d};z) &= G(\vec{a};\vec{b}-\vec{e_i};\vec{c};\vec{d};z)\\ C_iG(\vec{a};\vec{b};\vec{c};\vec{d};z) &= G(\vec{a};\vec{b};\vec{c}+\vec{e_i};\vec{d};z)\\ D_iG(\vec{a};\vec{b};\vec{c};\vec{d};z) &= G(\vec{a};\vec{b};\vec{c};\vec{d}+\vec{e_i};z)\end{aligned}$$

ここで、$\vec{e_i}$ は単位ベクトルである。これらの関係式を用いると、ある G 関数 $G(\vec{a};\vec{b};\vec{c};\vec{d};z)$ の公式が与えられると新たに多くの公式を得ることができる。このような関係を利用して新たに公式を生成するアルゴリズムは、超幾何関数に対する公式生成アルゴリズムを G 関数に拡張することで得られている。

3 データベース設計

本研究では、大別して 2 種類のデータベース (G 関数データベース、一般公式データベース) を作成した。Meijer の G 関数がすべての関数を網羅できるという保証はないので、それを補う目的で一般公式データベースを作成する。

3.1 概念スキーマの設計

- G 関数データベース
  G 関数データベースは以下のような概念スキーマによって設計されている。
  - 数学公式リレーション
    (公式番号、一般表現の式、G 関数表現の式、条件)
  - インデックスリレーション
    (公式番号、深さ、葉の数、キーワード数、キーワード)
  - G 関数リレーション
    (公式番号、$\vec{a}$ の要素、$\vec{b}$ の要素、$\vec{c}$ の要素、$\vec{d}$ の要素、z の値)

  数学公式リレーションについてはリレーショナルデータベースモデルに従い、インデックスリレーションと G 関数リレーションについては単一の行に複数列を持つような属性が存在するため、オブジェクトリレーショナルモデルに従って設計を行う。

- 一般公式データベース
  一般公式データベースは以下のような概念スキーマによって設計されている。
  - 数学公式リレーション
    (公式番号、積分前の式、積分後の式、条件)
  - インデックスリレーション
    (公式番号、深さ、葉の数、キーワード数、キーワード)

数学公式リレーションについてはリレーショナルデータベースモデルに従い、インデックスリレーションは単一の行に複数列を持つような属性が存在するため、オブジェクトリレーショナルモデルに従って設計を行う。

3.2 データベース検索方法

データベース検索は以下のような手順で行う。

1. 木構造より得られる特徴 (深さ、葉の数、キーワード、キーワード数) を取り出す。
2. 1 で得られた特徴をもとに一般公式データベースから検索を行う (インデックスリレーション)
3. 2 で見つからなければ G 関数データベースから検索を行う (インデックスリレーション)
   この作業により一般表現から G 関数表現に変換される
4. 検索の結果、唯一の公式が見つけられなければ次の方法でパターンマッチングを行う
   (a) 入力式の木構造の中で一番深いところにある OMA ノードを変数 OMV ノードに置き換える
   (b) 変更された木構造の特徴量を再計算し、インデックス検索を行う
   (c) 一致するものが見つからなければ (a) に戻る
   (d) これ以上変換できる OMA ノードがなくなったら処理を終了し、適合公式がないことを伝える。
5. 一般公式データベースの検索で唯一の公式が見つかっていれば終了

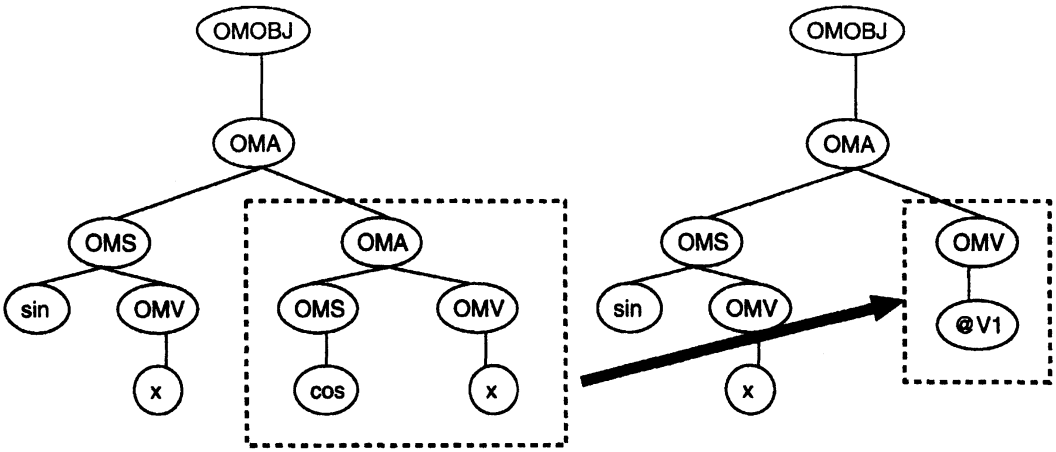

図 1: 任意の数式ノード (OMA) ノードの変換方法

6. G 関数データベースを用いた場合は、積分の処理を行い、特徴 ($\vec{a}$の要素、$\vec{b}$の要素、$\vec{c}$の要素、$\vec{d}$の要素、z の値) を求める

7. 再び G 関数データベースを用いて検索を行う (G 関数リレーション)
この作業により G 関数表現から一般表現に変換される

3.3 作成したデータベースのインデックス法の評価

作成したデータベースのインデックス法の評価を行う。インデックス法の評価の手段としては、再現率と適合率の 2 つの基準が存在する [8, 9]。これらの基準は、主に文献データベース検索システムにおけるインデックス法の評価を行うために用いられたが、ここでは数学公式データベースのインデックス法の評価に用いる。この場合、再現率は検索要求を満たす数学公式数に対する適合数学公式数の割合、適合率は検索された数学公式数に対する適合数学公式数の割合、と定義される。また、評価を行う際の条件は以下の通りである。

- 最終ステップであるパターンマッチイングを省略した結果である
- 検索漏れがないように人為的に十分注意をはらう
- データベースに実装したデータが対象である

公式検索において検索漏れを許したくない。公式に対する検索は、多くの場合唯一の公式を検索することを目的として実行される。このため、得られた検索結果に所望の公式が含まれていない場合は (ユーザーのミスでなければ)、インデックス法が根本的に誤りであったと結論づけなければならない。この理由により、インデックスの設計において検索漏れが発生しないように細心の注意を払った。その結果、検索実験において常に 100%の再現率が維持された。従って、着目しなければならないのは実際の検索においてどの程度の適合率が維持されたかという点のみである。適合率はあるキーで検索を行った場合、条件を満たす公式がどの程度検索されるかということであるが、以下のような結果となった。

1. 一般表現の式から G 関数表現の式に変換するとき

最も高い適合率 :100%　最も低い適合率 :10%　適合率の平均 :32%

2. G 関数表現の式から一般表現の式に変換するとき

適合率 :100%

この結果について考察すると、2 については、データベースに実装したデータが対象であることや G 関数はベクトルの値と z の値が固定されると数学的意味は一意になるという性質から 100%という結果になった。また、データベースにない公式は検索されないため、それを検索する場合は除いてある。1 と 2 両方の結果からは、一個の公式を得るために 2 個から 3 個のノイズを許すという基準は十分妥当な数字であると言える。また、検索により発生したノイズはパターンマッチングを適用することで除去され、唯一の公式を得ることができる。

4　システムの実際

4.1　システムの流れ

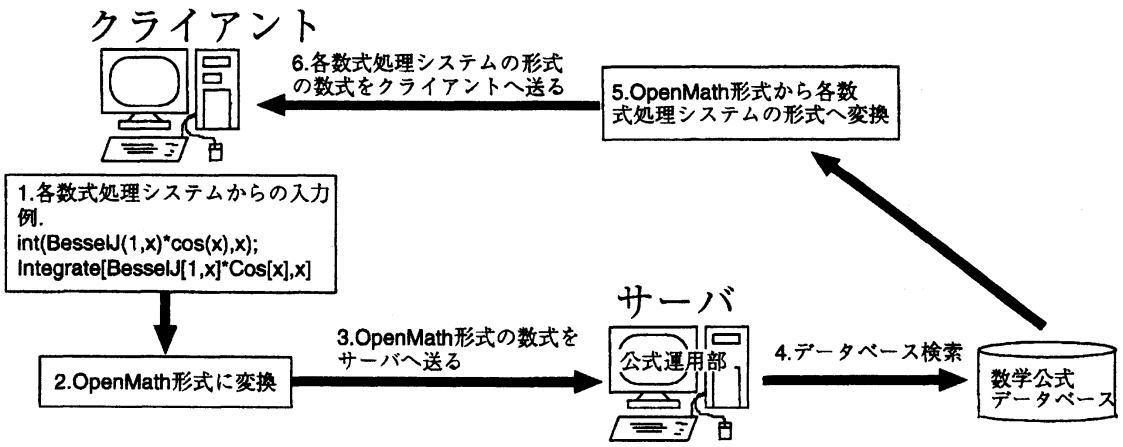

図 2: システムの流れ

システムの流れは以下の通りである。これを図 2 に示す。

1. クライアント側で各数式処理システム (Maple、Mathematica、Risa/Asir) から各数式処理システムの入力形式で入力を行う
2. 入力式を OpenMath 仕様によって標準化され XML 形式で書かれた数式に変換する
3. クライアントからサーバへ変換されたデータを送る
4. サーバで入力式の解析を行い、データベース検索を行う
5. 解は OpenMath 形式の数式で求められるため、各数式処理の形式に変換する
6. サーバからクライアント側の数式処理システムに数式を送り処理を終了する

4.2 問題点と解決方法

ここでは、本システムを実装する上で発生した問題点を提起し、その解決方法について述べる。

1. 積分の演算が存在しない数式処理システムで、作成したデータベースを利用する場合の問題点
   数式処理システムによっては積分の処理が実装されていないシステムが存在する。Risa/Asir などがこれに当たるが、各数式処理システム固有の積分の命令形態を新たに定義することで、データベースを利用可能にしている。

2. G 関数を用いて計算した場合

   (a) 問題点
   Meijer の G 関数を用いた積分は以下のような公式によって計算される。

$$\int G_{pq}^{mn}\left(z\left|\begin{matrix}a_1,\ldots,a_n,a_{n+1},\ldots,a_p\\ b_1,\ldots,b_m,b,_{m+1},\ldots,b_q\end{matrix}\right.\right)dz =$$
$$G_{p+1,q+1}^{m,n+1}\left(z\left|\begin{matrix}1,a_1+1,\ldots,a_n+1,a_{n+1}+1,\ldots,a_p+1\\ b_1+1,\ldots,b_m+1,0,b_m+1+1,\ldots,b_q+1\end{matrix}\right.\right) \quad (1)$$

   この式を見てもわかるように、積分前の n、p、q の値と積分後の n、p、q の値が異なり、積分後に数が増えているのがわかる。G 関数の性質上、数学的には同じ意味を示す数式でも G 関数では異なった表現となってしまう場合がある。

   i. 式 (1) を用いて計算した場合

$$\begin{aligned}\int \cos(x)dx &= \int \sqrt{\pi}G_{02}^{10}\left(\frac{x^2}{4}\left|\begin{matrix}\\ 0,\frac{1}{2}\end{matrix}\right.\right)dx\\ &= \sqrt{x}G_{02}^{10}\left(\frac{x^2}{4}\left|\begin{matrix}1\\ 1,0,\frac{1}{2}\end{matrix}\right.\right) = \sin(x)\end{aligned}$$

   ii. Shift Operator を用いて作成した場合

$$\sin(x) = \sqrt{x}G_{02}^{10}\left(\frac{x^2}{4}\left|\begin{matrix}\\ \frac{1}{2},0\end{matrix}\right.\right)$$

   このように、数学的意味では同じ $\sin(x)$ を表すのに、G 関数表現では異なった表現になってしまうことがある。

   (b) 解決方法
   この問題点の解決方法として考えられることは、まず Shift Operator を用いて公式を増やすことが考えられる。しかし、様々な場合にを想定しなければならず、この方法ではすべてを網羅することは不可能である。
   もう一つの方法として考えられるのは、変数を用いた公式を導入することである。G 関数には以下のような公式が知られている。

例 1 $$G_{13}^{11}\left(z\left|\begin{matrix}a\\ a-\frac{1}{2},a-1,a-1\end{matrix}\right.\right) = \frac{2z^{a-1}}{\sqrt{\pi}}Si(2\sqrt{z})$$

例 2 $$G_{22}^{21}\left(z\left|\begin{matrix}a,a\\ a-\frac{3}{2},a-\frac{1}{2}\end{matrix}\right.\right) = -4z^{a-1}E\left(-\frac{1}{z}\right)$$

G 関数の公式は例で示されたような変数を用いた公式が、現在までに約 1000 種知られている。これらの公式をデータベース化して、そのデータベースを用いることで、G 関数の公式から一般表現の式に変換を行っている。

5 まとめ

本研究では、数式処理システムに依存せずに利用可能な G 関数を用いた数学公式データベースの構築とその効率的なインデックス法の開発、及び、データベースを利用するための公式運用システムの構築を行った。その結果、積分の演算があるものにはその演算の強化を行うことができ、積分の演算がないものに関しては積分の命令を作成することによって、データベースを利用することにより積分の演算を行えるシステムを作成することができた。

今後の課題は、現在利用できる数式処理システムは Maple、Mathematica、Risa/Asir のみであるので、その他の数式処理システムでも扱えるようにすることである。

参 考 文 献

[1] The OpenMath Society
http://openmath.org

[2] The OpenMath Standard
http://www.nag.co.uk/projects/OpenMath/corecd

[3] The OpenMath Core Content Dictionaries
http://www.nag.co.uk/projects/OpenMath/corecd

[4] A.P.dnikov,Yu.A.Brychkov,O.I.Marichev,"Integral and Series,Volume 3: More Special Functions",Gordon and Breach Science Publishers,1986

[5] Adamchik v.s.,Marichev,The Algorithm for calculating integrals of hypergeometric type functions and its realization in reduce system,proceedings of ISSAC '90,pp.212-221

[6] kelly Rozch,Hypergeometric Function Representations,proceedings of ISSAC '96,pp.301-308

[7] kelly Rozch,Meijer G Function Representations,proceedings of ISSAC '97,pp.205-211

[8] 佐々木建昭、増永良文、阿部昭博、元吉文男、三枝義典、佐々木睦子、数式処理システム GAL における数学公式データベース、第２９回プログラミングシンポジウム、1988.1

[9] 三枝義典、阿部昭博、佐々木建昭、増永良文、佐々木睦子、数式処理システム GAL における数学公式データベースのインデキシング手法、電子情報通信学会論文誌 D-1Vol.J74-D-I No.8,pp.577-585

[10] 森永昌義、村上裕美、野田松太郎、数学公式データベースと G 関数、数理解析研究所講究録 1335「Computer Algebra - Algorithms,Implementations and Applications」、pp.27-1 - 27-8、2003